FFORMS: Feature-based forecast model selection

Abstract

Features of time series are useful in identifying suitable models for forecasting. We present a general framework, labelled FFORMS (Feature-based FORecast Model Selection), which selects forecast models based on features calculated from the time series. The FFORMS framework builds a mapping that relates the features of a time series to the "best" forecast model using a random forest. The framework is evaluated using time series from the M1 and M3 competitions and is shown to yield accurate forecasts comparable to several benchmarks and other commonly used automated approaches of time series forecasting. Furthermore, we explore what is happening under the hood of the FFORMS framework. This is accomplished using model-agnostic machine learning interpretability approaches. The analysis provides a valuable insight into how different features and their interactions affect the choice of forecast model.

Keywords: Algorithm selection problem, Time series, Random forest, Machine learning interpretability

1 Introduction

Forecasting is a key activity for any business to operate efficiently. The rapid advances in computing technologies have enabled businesses to keep track of large numbers of time series. Hence, it is becoming increasingly common to have to regularly forecast many millions of time series. For example, large scale businesses may be interested in forecasting sales, cost, and demand for their thousands of products across various locations, warehouses, etc. Technology companies such as Google collect many millions of daily time series such as web-click logs, web search counts, queries, revenues, number of users for different services, etc. Such large collections of time series require fast automated procedures generating accurate forecasts. The scale of these tasks has raised some computational challenges we seek to address by proposing a new fast algorithm for model selection and time series forecasting.

Two alternative strategies for generating such a large number of forecasts are: (1) to either use a single forecasting method across all the time series; or (2) to select an appropriate forecasting

method for each time series individually. It is highly unlikely that a single method will consistently outperform judiciously chosen competitor methods across all time series. We therefore reject the former strategy and focus on an approach for selecting an individual forecasting method for each time series.

Selecting the most appropriate model for forecasting a given time series can be challenging. Two of the most commonly used automated algorithms are the exponential smoothing (`ets`) algorithm of Hyndman et al. (2002) and the ARIMA (`auto.arima`) algorithm of Hyndman & Khandakar (2008). Both algorithms are implemented in the forecast package in R (R Core Team 2018; Hyndman et al. 2018). In this paradigm, a class of models is selected in advance, and many models within that class are estimated for each time series. The model with the smallest AICc value is chosen and used for forecasting. This approach relies on the expert judgement of the forecaster in first selecting the most appropriate class of models to use, as it is not usually possible to compare AICc values between model classes due to differences in the way the likelihood is computed, and the way initial conditions are handled.

An alternative approach, which avoids selecting a class of models *a priori*, is to use a simple "hold-out" test set; but then there is often insufficient data to draw a reliable conclusion. To overcome this drawback, time series cross-validation can be used (Hyndman & Athanasopoulos 2018); then models from many different classes may be applied, and the model with the lowest cross-validated MSE selected. However, this increases the computation time involved considerably (at least to order n^2 where n is the number of series to be forecast).

Clearly, there is a need for a fast and scalable algorithm to automate the process of selecting models with the aim of forecasting. There have been several recent studies on the use of meta-learning approaches to automate forecast model selection based on features computed from the time series (Shah 1997; Prudêncio & Ludermir 2004; Lemke & Gabrys 2010; Kück, Crone & Freitag 2016). A meta-learning approach provides systematic guidance on model selection based on knowledge acquired from historical data sets. The key idea is that forecast model selection is posed as a supervised learning task. Each time series in the metadata set is represented as a vector of features and labelled according to the best forecast model (for example, the model with the lowest mean absolute scaled error (MASE) over a test set). Then a meta-learner is trained to identify a suitable forecast model (usually a machine learning algorithm is used). In the era of big data, such an automated model selection process is necessary because the cost of invoking all possible forecast models is prohibitive. However, the existing literature suffers from limited answers to questions such as: How are features related to the property being

modelled? How do features interact with each other to identify a suitable forecast model? Which features contribute most to the classification process? Addressing such questions can enhance the understanding of the relations between features and model selection outcomes. To the best of our knowledge, a very limited effort has been made to understand how the meta-learners are making their decisions and what is really happening inside these complex model structures. Providing transparency will result, building trust in the prediction results of the meta-learner.

We propose a general meta-learning framework using features of the time series to select the class of models, or even the specific model, to be used for forecasting. We refer to this general framework as FFORMS (**F**eature-based **FOR**ecast-**M**odel **S**election). The forecast model selection process is carried out using a classification algorithm — we use the time series features as inputs, and the "best" forecasting model as the output. A random forest is used to model the relationship between features and best performing forecast model. The classifier is built using a large historical collection of time series, in advance of the forecasting task at hand. Hence, this is an "offline" procedure. The "online" process of generating forecasts only involves calculating the features of a time series and using the pre-trained classifier to identify the best forecasting model. Hence, generating forecasts only involves the estimation of a single forecasting model, with no need to estimate large numbers of models within a class, or to carry out a computationally-intensive cross-validation procedure. To fill the gaps in the literature, this paper makes a first step towards providing a comprehensive analysis of the relationship between time series features and forecast model selection using machine learning interpretability techniques.

In this article, we make the following contributions:

1. We explore the use of random forest algorithm to model the relationship between time series features and the 'best' forecast model. The motivation behind the use of random forest classifiers are: i) it can model complex interactions between features; ii) the modelling framework captures linear and non-linear effects of features through the averaging of large number of decision trees; iii) its robustness against over-fitting the training data; iv) it is easy to handle the problem of imbalanced classes; v) it is a fast approach compared to boosting algorithms and vi) it is fairly easy and straightforward to implement with available software. The contribution of this paper differs from previously published work related to meta-learning (Prudêncio & Ludermir 2004; Lemke & Gabrys 2010; Kück, Crone & Freitag 2016) in three ways: i) a more extensive collection of features is used (features to handle multiple seasonality in hourly and daily series), ii) the diversity of forecast models

considered class labels, and iii) the capability of handling high frequency data.

2. The second contribution of the paper is the exploration of what is happening under the hood of the FFORMS framework, leading to an understanding of the relationship between features of time series and the choice of forecast model selection using the FFORMS framework. We call it 'Peeking inside FFORMS'. We try to answer the following questions:

   i) **Which** features are the most important?
   ii) **Where** (overall classification or only within specific classes) are they important?
   iii) **How** are they important?
   iv) **When** and **how** are features linked to the prediction outcome?
   v) **When** and **how strongly** do features interact with other features?

2 Literature review

2.1 Time series features

Rather than work with the time series directly at the level of individual observations, we propose analysing time series via an associated "feature space". A time series feature is any measurable characteristic of a time series. For example, Figure 1 shows the time-domain representation of six time series taken from the M3 competition (Makridakis & Hibon 2000) while Figure 2 shows a feature-based representation of the same time series. Here only two features are considered: the strength of seasonality and the strength of trend, calculated based on the measures introduced by Wang, Smith-Miles & Hyndman (2009). Time series in the lower right quadrant of Figure 2 are non-seasonal but trended, while there is only one series with both high trend and high seasonality. We also see how the degree of seasonality and trend varies between series. Other examples of time series features include autocorrelation, spectral entropy and measures of self-similarity and nonlinearity. Fulcher & Jones (2014) identified 9000 operations to extract features from time series.

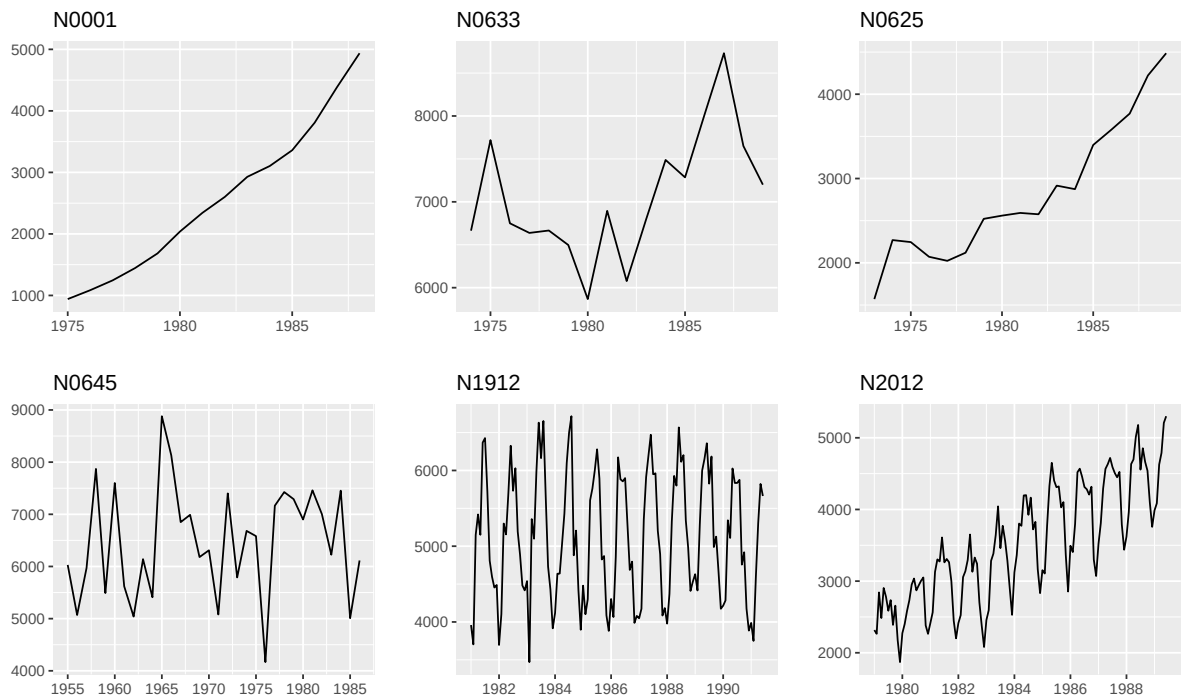

Figure 1: *Time-domain representation of time series*

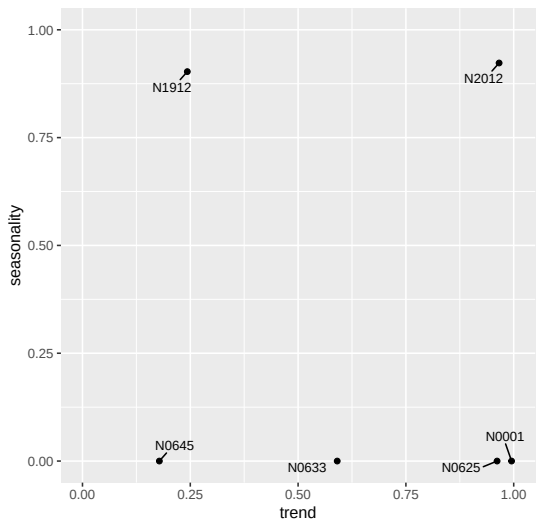

Figure 2: *Feature-based representation of time series*

2.2 Meta-learning for algorithm selection

John Rice was an early and strong proponent of the idea of meta-learning, which he called the algorithm selection problem (ASP) (Rice 1976). The term *meta-learning* started to appear with the emergence of the machine-learning literature. Rice's framework for algorithm selection is shown in Figure 3 and comprises four main components. The problem space, P, represents the data sets used in the study. The feature space, F, is the range of measures that characterize the problem space P. The algorithm space, A, is a list of suitable candidate algorithms which

can be used to find solutions to the problems in P. The performance metric, Y, is a measure of algorithm performance such as accuracy, speed, etc. A formal definition of the algorithm selection problem is given by Smith-Miles (2009), and repeated below.

> **Algorithm selection problem**. For a given problem instance $x \in P$, with features $f(x) \in F$, find the selection mapping $S(f(x))$ into algorithm space A, such that the selected algorithm $\alpha \in A$ maximizes the performance mapping $y(\alpha(x)) \in Y$.

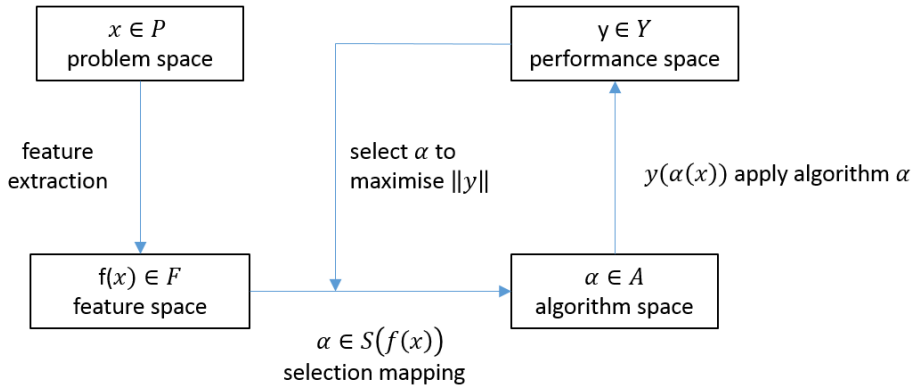

Figure 3: *Rice's framework for the Algorithm Selection Problem.*

The main challenge in ASP is to identify the selection mapping S from the feature space to the algorithm space. Even though Rice's framework articulates a conceptually rich framework, it does not specify how to obtain S. This gives rise to the meta-learning approach.

2.3 Forecast-model selection using meta-learning

Selecting models for forecasting can be framed according to Rice's ASP framework.

> **Forecast-model selection problem**. For a given time series $x \in P$, with features $f(x) \in F$, find the selection mapping $S(f(x))$ into the algorithm space A, such that the selected algorithm $\alpha \in A$ minimizes forecast accuracy error metric $y(\alpha(x)) \in Y$ on the test set of the time series.

Existing methods differ with respect to the way they define the problem space (A), the features (F), the forecasting accuracy measure (Y) and the selection mapping (S).

Collopy & Armstrong (1992) introduced 99 rules based on 18 features of time series, in order to make forecasts for economic and demographic time series. This work was extended by Armstrong (2001) to reduce human intervention.

Shah (1997) used the following features to classify time series: the number of observations, the

ratio of the number of turning points to the length of the series, the ratio of number of step changes, skewness, kurtosis, the coefficient of variation, autocorrelations at lags 1–4, and partial autocorrelations at lag 2–4. Casting Shah's work in Rice's framework, we can specify: $P = 203$ quarterly series from the M1 competition (Makridakis et al. 1982); $A = 3$ forecasting methods, namely simple exponential smoothing, Holt-Winters exponential smoothing with multiplicative seasonality, and a basic structural time series model; $Y =$ mean squared error for a hold-out sample. Shah (1997) learned the mapping S using discriminant analysis.

Prudêncio & Ludermir (2004) was the first paper to use the term "meta-learning" in the context of time series model selection. They studied the applicability of meta-learning approaches for forecast-model selection based on two case studies. Again using the notation above, we can describe their first case study with: A contained only two forecasting methods, simple exponential smoothing and a time-delay neural network; $Y =$ mean absolute error; F consisted of 14 features, namely length, autocorrelation coefficients, coefficient of variation, skewness, kurtosis, and a test of turning points to measure the randomness of the time series; S was learned using the C4.5 decision tree algorithm. For their second study, the algorithm space included a random walk, Holt's linear exponential smoothing and AR models; the problem space P contained the yearly series from the M3 competition (Makridakis & Hibon 2000); F included a subset of features from the first study; and Y was a ranking based on error. Beyond the task of forecast-model selection, they used the NOEMON approach to rank the algorithms (Kalousis & Theoharis 1999).

Lemke & Gabrys (2010) studied the applicability of different meta-learning approaches for time series forecasting. Their algorithm space A contained ARIMA models, exponential smoothing models and a neural network model. In addition to statistical measures such as the standard deviation of the de-trended series, skewness, kurtosis, length, strength of trend, Durbin-Watson statistics of regression residuals, the number of turning points, step changes, a predictability measure, nonlinearity, the largest Lyapunov exponent, and auto-correlation and partial-autocorrelation coefficients, they also used frequency domain based features. The feed forward neural network, decision tree and support vector machine approaches were considered to learn the mapping S.

Wang, Smith-Miles & Hyndman (2009) used a meta-learning framework to provide recommendations as to which forecast method to use to generate forecasts. In order to evaluate forecast accuracy, they introduced a new measure $Y =$ *simple percentage better (SPB)*, which calculates the forecasting accuracy of a method against the forecasting accuracy error of random walk model.

They used a feature set F comprising nine features: strength of trend, strength of seasonality, serial correlation, nonlinearity, skewness, kurtosis, self-similarity, chaos and periodicity. The algorithm space A included eight forecast-models, namely, exponential smoothing, ARIMA, neural networks and random walk model; while the mapping S was learned using the C4.5 algorithm for building decision trees. In addition, they used SOM clustering on the features of the time series in order to understand the nature of time series in a two-dimensional setting.

The set of features introduced by Wang, Smith-Miles & Hyndman (2009) was later used by Widodo & Budi (2013) to develop a meta-learning framework for forecast-model selection. The authors further reduced the dimensionality of time series by performing principal component analysis on the features.

More recently, Kück, Crone & Freitag (2016) proposed a meta-learning framework based on neural networks for forecast-model selection. Here, $P = 78$ time series from the NN3 competition were used to build the meta-learner. They introduced a new set of features based on forecasting errors. The average symmetric mean absolute percentage error was used to identify the best forecast-models for each series. They classify their forecast-models in the algorithm space A, comprising single, seasonal, seasonal-trend and trend exponential smoothing. The mapping S was learned using a feed-forward neural network. Further, they evaluated the performance of different sets of features for forecast-model selection.

3 Methodology

Our proposed FFORMS framework, presented in Figure 4, builds on this preceding research. The offline and online phases are shown in blue and red respectively. A classification algorithm (the meta-learner) is trained during the offline phase and is then used to select an appropriate forecast model for a new time series in the online phase. Furthermore, we use machine learning interpretability tools to gain insights into what is happening under the hood of the FFORMS framework. We refer to this process as "peeking inside FFORMS" which is shown in yellow.

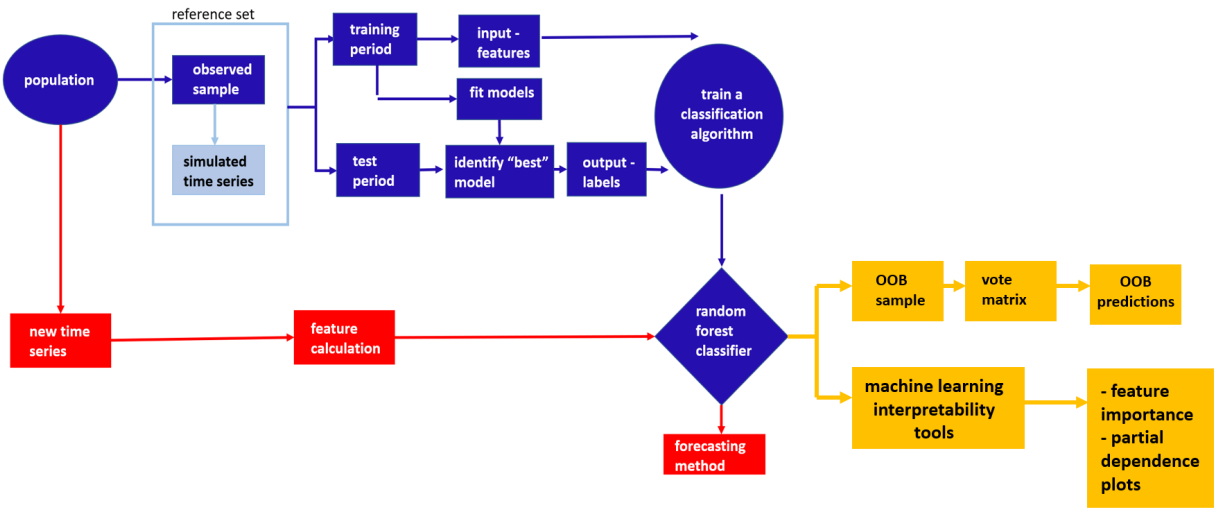

Figure 4: *FFORMS (Feature-based FORecast-Model Selection) framework. The offline phase is shown in blue, the online phase in red and the peeking inside FFORMS is shown in yellow.*

In order to train our classification algorithm, we need a large collection of time series which are similar to those we will be forecasting. We assume that we have an essentially infinite population of time series, and we take a sample of them in order to train the classification algorithm denoted as the "observed sample". The new time series we wish to forecast can be thought of as additional draws from the same population. Hence, any conclusions made from the classification framework refer only to the population from which the sample has been selected. We may call this the "target population" of time series. It is important to have a well-defined target population to avoid misapplying the classification rules. In practice, we may wish to augment the set of observed time series by simulating new time series similar to those in the assumed population (we provide details and discussion in Section 3.1 that follows). We denote the total collection of time series used for training the classifier as the "reference set". Each time series within the reference set is split into a training period and a test period. From each training period we compute a range of time series features, and fit a selection of candidate models. The calculated features form the input vector to the classification algorithm. Using the fitted models, we generate forecasts and identify the "best" model for each time series based on a forecast error measure (e.g., MASE) calculated over the test period. The models deemed "best" form the output labels for the classification algorithm. The pseudo code for our proposed framework is presented in Algorithm 1 below. In the following sections, we briefly discuss

aspects of the offline phase.

Algorithm 1 The FFORMS framework - Forecasting based on meta-learning.

Offline phase - train the classifier
Given:
$O = \{x_1, x_2, \ldots, x_n\}$: the collection of n observed time series.
L : the set of class labels (e.g. ARIMA, ETS, SNAIVE).
F : the set of functions to calculate time series features.
$nsim$: number of series to be simulated.
B : number of trees in the random forest.
k : number of features to be selected at each node.
Output:
FFORMS classifier

Prepare the reference set
For $i = 1$ to n:
1: Fit ARIMA and ETS models to x_i.
2: Simulate $nsim$ time series from each model in step 1.
3: The time series in O and simulated time series in step 2 form the reference set $R = \{x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_N\}$ where $N = n + nsim$.

Prepare the meta-data
For $j = 1$ to N:
4: Split x_j into a training period and test period.
5: Calculate features F based on the training period.
6: Fit L models to the training period.
7: Calculate forecasts for the test period from each model.
8: Calculate forecast error measure over the test period for all models in L.
9: Select the model with the minimum forecast error.
10: Meta-data: input features (step 5), output labels (step 9).

Train a random forest classifier
11: Train a random forest classifier based on the meta-data.
12: Random forest: the ensemble of trees $\{T_b\}_1^B$.

Online phase - forecast a new time series
Given:
FFORMS classifier from step 12 .
Output:
class labels for new time series x_{new}.
13: For x_{new} calculate features F.
14: Let $\hat{C}_b(x_{new})$ be the class prediction of the bth random forest tree. Then class label for x_{new} is $\hat{C_{rf}}(x_{new}) = majorityvote\{\hat{C}_b(x_{new})\}_1^B$.

3.1 Augmenting the observed sample with simulated time series

In practice, we may wish to augment the set of observed time series by simulating new time series similar to those in the assumed population. This process may be useful when the number of observed time series is too small to build a reliable classifier. Alternatively, we may wish

to add more of some particular types of time series to the reference set in order to get a more balanced sample for the classification. In order to produce simulated series that are similar to those in the population, we consider two classes of data generating processes: exponential smoothing models and ARIMA models. Using the automated `ets` and `auto.arima` algorithms (Hyndman et al. 2018) we identify models, based on model selection criteria (such as AICc) and simulate multiple time series from the selected models within each model class. Assuming the models produce data that are similar to the observed time series ensures that the simulated series are similar to those in the population. As this is done in the offline phase of the algorithm the computational time in producing these simulated series is of no real consequence.

3.2 Input: features

Our proposed FFORMS algorithm requires features that enable identification of a suitable forecast model for a given time series. Therefore, the features used should capture the dynamic structure of the time series, such as trend, seasonality, autocorrelation, nonlinearity, heterogeneity, and so on. Furthermore, interpretability, robustness to outliers, scale and length independence should also be considered when selecting features for this classification problem. A comprehensive description of the features used in the experiment is presented in Table 1.

A feature of the proposed FFORMS framework is that its online phase is fast compared to the time required for implementing a typical model selection or cross-validation procedure. Therefore, we consider only features that can be computed rapidly, as this computation must be done during the online phase.

3.3 Output: labels

The task of the classification algorithm is to identify the "best" forecasting method for a given time series. The candidate models considered as labels will depend on the observed time series. For example, if we have only non-seasonal time series, and no chaotic features, we may wish to restrict the models to white noise, random walk, ARIMA and ETS processes. We fit the corresponding models outlined in Table 2 to each series in the reference set.

Each candidate model considered is estimated strictly on the training period of each series in the reference set. Forecasts are then generated for the test period over which the chosen forecast accuracy measure is calculated. The model with the lowest forecast error measure over the test period is deemed "best".

This step is the most computationally intensive and time-consuming, as each candidate model has to be applied on each series in the reference set. However, as this task is done during the

Table 1: *Time series features*

	Feature	Description	Y	Q/M	W	D/H
1	T	length of time series	✓	✓	✓	✓
2	trend	strength of trend	✓	✓	✓	✓
3	seasonality_q	strength of quarterly seasonality	-	✓	-	-
4	seasonality_m	strength of monthly seasonality	-	✓	-	-
5	seasonality_w	strength of weekly seasonality	-	-	✓	✓
6	seasonality_d	strength of daily seasonality	-	-	-	✓
7	seasonality_y	strength of yearly seasonality	-	-	-	✓
8	linearity	linearity	✓	✓	✓	✓
9	curvature	curvature	✓	✓	✓	✓
10	spikiness	spikiness	✓	✓	✓	✓
11	e_acf1	first ACF value of remainder series	✓	✓	✓	✓
12	stability	stability	✓	✓	✓	✓
13	lumpiness	lumpiness	✓	✓	✓	✓
14	entropy	spectral entropy	✓	✓	✓	✓
15	hurst	Hurst exponent	✓	✓	✓	✓
16	nonlinearity	nonlinearity	✓	✓	✓	✓
17	alpha	ETS(A,A,N) $\hat{\alpha}$	✓	✓	✓	-
18	beta	ETS(A,A,N) $\hat{\beta}$	✓	✓	✓	-
19	hwalpha	ETS(A,A,A) $\hat{\alpha}$	-	✓	-	-
20	hwbeta	ETS(A,A,A) $\hat{\beta}$	-	✓	-	-
21	hwgamma	ETS(A,A,A) $\hat{\gamma}$	-	✓	-	-
22	ur_pp	test statistic based on Phillips-Perron test	✓	-	-	-
23	ur_kpss	test statistic based on KPSS test	✓	-	-	-
24	y_acf1	first ACF value of the original series	✓	✓	✓	✓
25	diff1y_acf1	first ACF value of the differenced series	✓	✓	✓	✓
26	diff2y_acf1	first ACF value of the twice-differenced series	✓	✓	✓	✓
27	y_acf5	sum of squares of first 5 ACF values of original series	✓	✓	✓	✓
28	diff1y_acf5	sum of squares of first 5 ACF values of differenced series	✓	✓	✓	✓
29	diff2y_acf5	sum of squares of first 5 ACF values of twice-differenced series	✓	✓	✓	✓
30	sediff_acf1	ACF value at the first lag of seasonally-differenced series	-	✓	✓	✓
31	sediff_seacf1	ACF value at the first seasonal lag of seasonally-differenced series	-	✓	✓	✓
32	sediff_acf5	sum of squares of first 5 autocorrelation coefficients of seasonally-differenced series	-	✓	✓	✓
33	seas_pacf	partial autocorrelation coefficient at first seasonal lag	-	✓	✓	✓
34	lmres_acf1	first ACF value of residual series of linear trend model	✓	-	-	-
35	y_pacf5	sum of squares of first 5 PACF values of original series	✓	✓	✓	✓
36	diff1y_pacf5	sum of squares of first 5 PACF values of differenced series	✓	✓	✓	✓
37	diff2y_pacf5	sum of squares of first 5 PACF values of twice-differenced series	✓	✓	✓	✓

offline phase of the FFORMS framework, the time involved and computational cost associated are of no real significance and are completely controlled by the user.

3.4 Train a FFORMS meta-learner

A random forest algorithm is used to train the meta-learner. In this work, we have used the randomForest package (Liaw & Wiener 2002; Breiman et al. 2018) which implements the Fortran code for random forest classification by Breiman & Cutler (2004). To determine the class label for a new instance, features are calculated and passed down the trees. Then each tree gives a prediction and the majority vote over all individual trees leads to the final decision.

The random forest (RF) algorithm is highly sensitive to class imbalance (Breiman 2001), and our

Table 2: *Class labels*

Class label	Description	Y	Q/M	W	D/H
wn	white noise process	✓	✓	✓	✓
ARMA	AR, MA, ARMA processes	✓	✓	✓	-
ARIMA	ARIMA process	✓	✓	✓	-
SARIMA	seasonal ARIMA	-	✓	✓	-
rwd	random walk with drift	✓	✓	✓	✓
rw	random walk	✓	✓	✓	✓
theta	standard theta method	✓	✓	✓	✓
stlar		-	✓	✓	✓
ETS_NTNS	ETS without trend and seasonal components	✓	✓	-	-
ETS_T	ETS with trend component and without seasonal component	✓	✓	-	-
ETS_DT	ETS with damped trend component and without seasonal component	✓	✓	-	-
ETS_TS	ETS with trend and seasonal component	-	✓	-	-
ETS_DTS	ETS with damped trend and seasonal component	-	✓	-	-
ETS_S	ETS with seasonal component and without trend component	-	✓	-	-
snaive	seasonal naive method	-	✓	✓	✓
tbats	TBATS forecasting	-	✓	✓	✓
nn	neural network time series forecasts	✓	✓	✓	✓
mstlets		-	-	✓	✓
mstlarima		-	-	-	✓

reference set is unbalanced: some classes contain significantly more cases than other classes. The degree of class imbalance is reduced to some extent by augmenting the observed sample with the simulated time series. We consider three approaches to address the class imbalance in the data: (1) incorporating class priors into the RF classifier; (2) using the balanced RF algorithm introduced by Chen, Liaw & Breiman (2004); and (3) re-balancing the reference set with down-sampling. In down-sampling, the size of the reference set is reduced by down-sampling the larger classes so that they match the smallest class in size; this potentially discards some useful information. Comparing the results, the balanced RF algorithm and RF with down-sampling did not yield satisfactory results. We therefore only report the results obtained by the RF built on unbalanced data (RF-unbalanced) and the RF with class priors (RF-class priors). The RF algorithms are implemented by the randomForest R package (Liaw & Wiener 2002; Breiman et al. 2018). The class priors are introduced through the option `classwt`. We use the reciprocal of class size as the class priors. The number of trees `ntree` is set to 1000, and the number of randomly selected features `f` is set to be one third of the total number of features available.

Our aim is different from most classification problems in that we are not interested in accurately predicting the class, but in finding the best possible forecast model. It is possible that two models produce almost equally accurate forecasts, and therefore it is not important whether the classifier picks one model over the other. Therefore we report the forecast accuracy obtained from the FFORMS framework, rather than the classification accuracy.

4 Peeking inside FFORMS

The main objective of this paper is to explore the nature of the relationship between features and forecast model selection learned by the FFORMS framework. More specifically, it is to identify which of the features are important for model predictions and how different features and their interactions led to the different choices. First, we explore the global role of features in the choice of different forecast model selection. Global interpretability evaluates the behaviour of a model on an entire data set. The global perspective of model interpretation helps users to understand the overall modelled relationship between features and the FFORMS outcome. In the following subsections, we provide a description of tools we use to explore the global perspective of the FFORMS meta-learners.

4.1 Visualise patterns learned by the meta-learner

A useful by-product of random forest model is out-of-bag (OOB) observations. OOB observations are the observations not included in building a given tree. In general, each tree uses only around one third of observations in its construction. Further, each tree is grown based on different bootstrap samples; hence, each tree has a different set of OOB observations. We use a vote matrix calculated based on OOBs to visualise patterns learned by the random forest. The vote matrix ($N \times P$; N is the total number of observations; P is the number of classes) contains the proportion of times each observation was classified to each class based on OOB observations.

4.2 Feature importance

Let $\mathcal{P} = \{(\mathbf{f_i}, z_i)\}_{i=1}^{N}$ be the historical data set we use to train a classifier. Consider a d-dimensional feature vector $\mathbf{F} = (F_1, F_2, ..., F_d)$ and a dependent variable, the best forecasting method for each series Z. Let $\mathcal{G}$ be the unknown relationship between $\mathbf{F}$ and Z. **Zhao** call this 'law of nature'. Inside the FFORMS framework, the random forest algorithm tries to learn this relationship using the historical data we provided. We denote the predicted function as g.

jiang2002 explain variable importance under three different views: i) causality (change in the value of Z for an increase or decrease in the value of F_i), ii) contribution of F_i based on out-of-sample prediction accuracy and iii) face value of F_i on prediction function g. For example, in linear regression models estimated coefficients of each predictor can be considered a measure of variable importance. See **jiang2002** for comparable face value interpretation for machine learning models. In this paper we use the first two notions of variable importance. Partial dependency functions and individual conditional expectation curves are used to explore the 'causality' notion of variable importance, while mean decrease in Gini coefficient and permutation-based

variable importance are used to capture the second notion of variable importance (feature contribution to the predictive accuracy) (**Zhao**). We introduce each of these variable importance measures below.

4.2.1 Mean decrease in Gini coefficient

Mean decrease in Gini coefficient is a measure of how each feature contributes to the homogeneity of the nodes and leaves in the resulting random forest proposed by Breiman (2001).

4.2.2 Permutation-based variable importance measure

The permutation-based variable importance introduced by Breiman (2001) measures the prediction strength of each feature. This measure is calculated based on the out-of-bag (OOB) observations. The calculation of variable importance is formalised as follows: Let $\bar{\mathcal{B}}^{(k)}$ be the OOB sample for a tree k, with $k \in \{1, ..., ntree\}$, where $ntree$ is the number of trees in the random forest. Then the variable importance of variable F_j in k^{th} tree is:

$$VI^{(k)}(F_j) = \frac{\sum_{i \in \bar{\mathcal{B}}^{(k)}} I(\gamma_i = \gamma^k_{i,\pi_j})}{|\bar{\mathcal{B}}^{(k)}|} - \frac{\sum_{i \in \bar{\mathcal{B}}^{(k)}} I(\gamma_i = \gamma^k_i)}{|\bar{\mathcal{B}}^{(k)}|},$$

where γ^k_i denotes the predicted class for the ith observation before permuting the values of F_j and γ^k_{i,π_j} is the predicted class for the ith observation after permuting the values of F_j. The overall variable importance score is calculated as:

$$VI(F_j) = \frac{\sum_{k=1}^{ntree} VI^{(k)}(F_j)}{ntree}.$$

Permutation-based variable importance measures provide a useful starting point for identifying the relative influence of features on the predicted outcome. However, they provide little indication of the nature of the relationship between the features and model outcome. To gain further insights into the role of features inside the FFORMS framework we use the partial dependence plot (PDP) introduced by **friedman2008predictive**.

4.2.3 Partial dependence plot (PDP) and variable importance measure based on PDP

Partial dependence plots can be used to graphically examine how each feature is related to the model prediction while accounting for the average effect of other features in the model. Let F_s be the subset of features we want to examine for partial dependencies and F_c be the remaining

set of features in F. Then g_s, the partial dependence function on F_s is defined as

$$g_s(F_s) = E_{f_c}[g(f_s, F_c)] = \int g(f_s, f_c) dP(f_c).$$

In practice, PDP can be estimated from a training data set as

$$\bar{g}_s(f_s) = \frac{1}{n} \sum_{i=1}^{n} g(f_s, F_{iC}),$$

where n is the number of observations in the training data set. A partial dependency curve can be created by plotting the pairs of $\{(f_s^k, \bar{g}_s(f_{sk}))\}_{k=1}^m$ defined on a grid of points $\{f_{s1}, f_{s2}, \ldots, f_{sm}\}$ based on F_s. The FFORMS framework has treated the forecast model selection problem as a classification problem. Hence, in this paper partial dependency functions display the probability of a certain class occurring given different values of the feature F_s.

Greenwell2018 introduce a variable importance measure based on the partial dependency curves. The idea is to measure the 'flatness' of partial dependence curves for each feature. A feature with a PDP curve that is flat, relative to the other features, does not have much influence on the predicted value. The flatness of the curve is measured using the standard deviation of the values $\{\bar{g}_s(f_{sk})\}_{k=1}^m$.

4.2.4 Individual conditional expectation (ICE) curves and variable importance measure based on ICE curves

While partial dependency curves are useful in understanding the estimated relationship between features and the predicted outcome in the presence of substantial interaction between features, they can be misleading. **goldstein2015peeking** propose individual conditional expectation (ICE) curves to address this issue. Instead of averaging $g(f_s, F_{iC})$ over all observations in the training data, an ICE curve generates the individual response curves by plotting the pairs $\{(f_s^k, g(f_{sk}, F_{iC}))\}_{k=1}^m$ defined on grid of points $\{f_{s1}, f_{s2}, \ldots, f_{sm}\}$ based on F_s. In other words, the partial dependency curve is simply the average of all the ICE curves.

This method is similar to the PDP-based variable importance scores above, but is based on measuring the 'flatness' of the individual conditional expectation curves. We calculate standard deviations of each ICE curve. We then compute an ICE-based variable importance score, which is simply the average of all the standard deviations. A higher value indicates a higher degree of interactivity with other features.

4.2.5 Ranking of features based on feature importance measures

To identify important class-specific features we rank features in three different ways: i) based on permutation-based variable importance, ii) based on partial dependence functions and iii) based on ICE curves. We consider 25 features for yearly data. The feature that shows the highest importance is ranked 25, the second highest is ranked 24, and so on. Finally, for each feature a mean rank is calculated based on the rankings of the three measures. Similarly, the overall feature importance is evaluated based on the permutation-based variable importance measure and the Gini coefficient-based feature importance measure.

4.3 Relationship between most important features and the choice of forecast model selection

The partial-dependence curves, along with their confidence intervals, are used to visualise the relationship between top features and the choice of forecast model selection.

4.4 Assessment of interaction effect

Friedman's H-statistic (**friedman2008predictive**) is used to test the presence of interaction between all possible pairs of features. This statistic is computed based on the partial dependence functions. For two-way interaction between two specific variables Y_j and Y_k, Friedman's H-statistic is defined as follows:

$$H_{jk}^2 = \sum_{i=1}^{n} [\bar{g}_s(f_{ij}, f_{jk}) - \bar{g}_s(f_{ij}) - \bar{g}_s(f_{ik})]^2 / \sum_{i=1}^{n} \bar{g}_s^2(f_{ij}, f_{jk}).$$

Friedman's H-statistic measures the fraction of variance of two-variable partial dependency, $\bar{g}_s(f_{ij}, f_{jk})$ not captured by the sum of the respective individual partial dependencies, $\bar{g}_s(f_{ij}) + \bar{g}_s(f_{ik})$. In addition to Friedman's H-statistic, we use the PDP of two variables to visualise the interaction effects. Similarly, Friedman's H-statistic for testing whether a specific feature interacts with any other feature can be estimated by

$$H_j^2 = \sum_{i=1}^{n} [g(\boldsymbol{f_i}) - \bar{g}_s(f_{ij}) - \bar{g}_s(f_{i/j})]^2 / \sum_{i=1}^{n} g^2(\boldsymbol{f_i}).$$

Note that the, PD plots, ICE curves and PD- and ICE-associated measures and Friedman's H-statistic are computationally intensive to compute, especially when there are a large number of observations in the training set. Hence, in our experiments ICE- and PDP-based variable importance measures are computed based on the subset of randomly selected training examples.

5 Results

5.1 Application to M4-competition data

Table 3: *The performance of FFORMS on the M4 competition data based on point forecasts (based on MASE) and prediction intervals (based on MSIS)*

Point Forecasts (Mean Absolute Scaled Error (MASE))						
	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly
FFORMS	3.17	1.20	0.98	2.31	3.57	0.84
auto.arima	3.40	1.17	0.93	2.55	-	-
ets	3.44	1.16	0.95	-	-	-
theta	3.37	1.24	0.97	2.64	3.33	1.59
rwd	3.07	1.33	1.18	2.68	3.25	11.45
rw	3.97	1.48	1.21	2.78	3.27	11.60
nn	4.06	1.55	1.14	4.04	3.90	1.09
stlar	-	2.02	1.33	3.15	4.49	1.49
snaive	-	1.66	1.26	2.78	24.46	2.86
tbats	-	1.19	1.05	2.49	3.27	1.30
wn	13.42	6.50	4.11	49.91	38.07	11.68
mstlarima	-	-	-	-	3.84	1.12
mstlets	-	-	-	-	3.73	1.23
combination (mean)	4.09	1.58	1.16	6.96	7.94	3.93
M4-1st	**2.98**	1.12	**0.88**	2.36	3.45	0.89
M4-2nd	3.06	**1.11**	0.89	**2.11**	3.34	**0.81**
M4-3rd	3.13	1.12	0.91	2.16	**2.64**	0.87
Prediction Intervals (Mean Scaled Interval Score (MSIS))						
FFORMS	39.79	11.24	9.82	**20.84**	36.36	8.0
M4-1st	**23.89**	**8.55**	**7.20**	22.03	**26.28**	**7.92**
M4-2nd	27.47	9.38	8.65	21.53	34.37	18.50
M4-3rd	not submitted					
naive	56.55	14.07	12.30	26.35	32.55	71.24

Note: Bold signifies the best performing method.

Table 3 shows the performance of the FFORMS approach on the M4 competition data. The point forecasts and prediction intervals are evaluated based on the test period of each series. The point forecasts are evaluated based on the MASE computed for each forecast horizon, and then by averaging the MASE errors across all series corresponding to each frequency category. Similarly, MSIS is used to evaluate the prediction intervals. The results are compared against several benchmarks and the top three places of the M4 competition. The top-ranking methods of the M4 competition are based on some kind of combination approach. The first ranking method is based on a hybrid approach that produces forecasts based on both ETS and machine learning approaches. The second and third places are based on a combination of nine and seven different forecast models. The second and third approaches are time-consuming because forecasts from all candidate models must be computed. The results of the FFORMS approach are based on individual forecasts. According to the Table 3, we can see the FFORMS approach achieved comparable performance in a much more cost and time-effective way. Based on the FFORMS approach, we provided an entry to the M4 competition. After submission we found a bug in our implementation of hourly data, hence only the hourly data results are different from the published results of the M4 competition. The yearly, quarterly, monthly, weekly and daily result

are same as the published results. This completes the evaluation of the FFORMS framework. The main question we have now is: 'Can we trust the FFORMS framework if we do not know how it works?' In the next section we present results showing what is happening inside the FFORMS framework.

5.2 What is happening under the hood of FFORMS?

5.2.1 Visualisation of meta-learner's understanding of similarity and dissimilarity between forecast models

5.2.2 Time series without seasonal component: Yearly

5.2.3 Time series with a single seasonal component: Monthly

5.2.4 Time series with multiple seasonal components: Hourly

6 Discussion and conclusions

In this paper we have proposed a novel framework for forecast-model selection using meta-learning based on time series features. Our proposed FFORMS algorithm uses the knowledge of the past performance of candidate forecast models on a collection of time series in order to identify the best forecasting method for a new series. We have shown that the method almost always performs better than common benchmark methods, and in many cases better than the best-performing methods from both the M1 and the M3 forecasting competitions. Although we have illustrated the method using the M1 and M3 competition data, the framework is general and can be applied to any large collection of time series.

A major advantage of the FFORMS framework is that the classifier is trained offline and selecting a forecasting model for a new time series is as fast as calculating a set of features and passing these to the pretrained classifier. Therefore, unlike traditional model selection strategies or cross-validation processes, our proposed framework can be used to accurately forecast very large collections of time series requiring almost instant forecasts.

In addition to our new FFORMS framework, we have also introduced a simple set of time series features that are useful in identifying the "best" forecast method for a given time series, and can be computed rapidly. We will leave to a later study an analysis of which of these features are the most useful, and how our feature set could be reduced further without the loss of forecast accuracy.

For real-time forecasting, our framework involves only the calculation of features, the selection of a forecast method based on the FFORMS random forest classifier, and the calculation of the

forecasts from the chosen model. None of these steps involve substantial computation, and they can be easily parallelised when forecasting for a large number of new time series. For future work, we will explore the use of other classification algorithms within the FFORMS algorithm, and test our approach on several other large collections of time series.

Appendix A: Definition of features

Length of time series

The appropriate forecast method for a time series depends on how many observations are available. For example, shorter series tend to need simple models such as a random walk. On the other hand, for longer time series, we have enough information to be able to estimate a number of parameters. For even longer series (over 200 observations), models with time-varying parameters give good forecasts as they help to capture the changes of the model over time.

Features based on an STL-decomposition

The strength of trend, strength of seasonality, linearity, curvature, spikiness and first autocorrelation coefficient of the remainder series, are calculated based on a decomposition of the time series. Suppose we denote our time series as $y_1, y_2, \ldots, y_T$. First, an automated Box-Cox transformation (Guerrero 1993) is applied to the time series in order to stabilize the variance and to make the seasonal effect additive. The transformed series is denoted by y_t^*. For quarterly and monthly data, this is decomposed using STL (Cleveland, Cleveland & Terpenning 1990) to give $y_t^* = T_t + S_t + R_t$, where T_t denotes the trend, S_t denotes the seasonal component, and R_t is the remainder component. For non-seasonal data, Friedman's super smoother (Friedman 1984) is used to decompose $y_t^* = T_t + R_t$, and $S_t = 0$ for all t. The de-trended series is $y_t^* - T_t = S_t + R_t$, the deseasonalized series is $y_t^* - S_t = T_t + R_t$..

The strength of trend is measured by comparing the variance of the deseasonalized series and the remainder series (Wang, Smith-Miles & Hyndman 2009):

$$\text{Trend} = \max\left[0, 1 - \text{Var}(R_t)/\text{Var}(T_t + R_t)\right].$$

Similarly, the strength of seasonality is computed as

$$\text{Seasonality} = \max\left[0, 1 - \text{Var}(R_t)/\text{Var}(S_t + R_t)\right].$$

The linearity and curvature features are based on the coefficients of an orthogonal quadratic

regression

$$T_t = \beta_0 + \beta_1\phi_1(t) + \beta_2\phi_2(t) + \varepsilon_t,$$

where $t = 1, 2, \ldots, T$, and ϕ_1 and ϕ_2 are orthogonal polynomials of orders 1 and 2. The estimated value of β_1 is used as a measure of linearity while the estimated value of β_2 is considered as a measure of curvature. These features were used by Hyndman, Wang & Laptev (2015). The linearity and curvature depend on the the scale of the time series. Therefore, the time series are scaled to mean zero and variance one before these two features are computed.

The spikiness feature is useful when a time series is affected by occasional outliers. Hyndman, Wang & Laptev (2015) introduced an index to measure spikiness, computed as the variance of the leave-one-out variances of r_t.

We compute the first autocorrelation coefficient of the remainder series, r_t.

Stability and lumpiness

The features "stability" and "lumpiness" are calculated based on tiled windows (i.e., they do not overlap). For each window, the sample mean and variance are calculated. The stability feature is calculated as the variance of the means, while lumpiness is the variance of the variances. These were first used by Hyndman, Wang & Laptev (2015).

Spectral entropy of a time series

Spectral entropy is based on information theory, and can be used as a measure of the forecastability of a time series. Series that are easy to forecast should have a small spectral entropy value, while very noisy series will have a large spectral entropy. We use the measure introduced by Goerg (2013) to estimate the spectral entropy. It estimates the Shannon entropy of the spectral density of the normalized spectral density, given by

$$H_s(y_t) := -\int_{-\pi}^{\pi} \hat{f}_y(\lambda) \log \hat{f}_y(\lambda) d\lambda,$$

where $\hat{f}_y$ denotes the estimate of the spectral density introduced by Nuttall & Carter (1982). The R package ForeCA (Goerg 2016) was used to compute this measure.

Hurst exponent

The Hurst exponent measures the long-term memory of a time series. The Hurst exponent is given by $H = d + 0.5$, where d is the fractal dimension obtained by estimating a ARFIMA$(0, d, 0)$ model. We compute this using the maximum likelihood method (Haslett & Raftery 1989) as implemented in the fracdiff package (Fraley 2012). This measure was also used in Wang,

Smith-Miles & Hyndman (2009).

Nonlinearity

To measure the degree of nonlinearity of the time series, we use statistic computed in Terasvirta's neural network test for nonlinearity (Teräsvirta, Lin & Granger 1993), also used in Wang, Smith-Miles & Hyndman (2009). This takes large values when the series is nonlinear, and values around 0 when the series is linear.

Parameter estimates of an ETS model

The ETS (A,A,N) model (Hyndman et al. 2008) produces equivalent forecasts to Holt's linear trend method, and can be expressed as follows:

$$
\begin{aligned}
y_t &= \ell_{t-1} + b_{t-1} + \varepsilon_t \\
\ell_t &= \ell_{t-1} + b_{t-1} + \alpha\varepsilon_t \\
b_t &= b_{t-1} + \beta\varepsilon_t,
\end{aligned}
$$

where α is the smoothing parameter for the level, and β is the smoothing parameter for the trend. We include the parameter estimates of both α and β in our feature set for yearly time series. These indicate the variability in the level and slope of the time series.

The ETS(A,A,A) model (Hyndman et al. 2008) produces equivalent forecasts to Holt-Winters' additive method, and can be expressed as follows:

$$
\begin{aligned}
y_t &= \ell_{t-1} + b_{t-1} + s_{t-m} + \varepsilon_t \\
\ell_t &= \ell_{t-1} + b_{t-1} + s_{t-m} + \alpha\varepsilon_t \\
b_t &= b_{t-1} + \beta\varepsilon_t, \\
s_t &= s_{t-m} + \gamma\varepsilon_t,
\end{aligned}
$$

where γ is the smoothing parameter for the seasonal component, and the other parameters are as above. We include the parameter estimates of α, β and γ in our feature set for monthly and quarterly time series. The value of γ provides a measure for the variability of the seasonality of a time series.

Unit root test statistics

The Phillips-Perron test is based on the regression $y_t = c + \alpha y_{t-1} + \varepsilon_t$. The test statistic we use as a feature is the usual "Z-alpha" statistic with the Bartlett window parameter set to the integer value of $4(T/100)^{0.25}$ (Pfaff 2008). This is the default value returned from the `ur.pp()` function

in the `urca` package (Pfaff, Zivot & Stigler 2016).

The KPSS test is based on the regression $y_t = c + \delta t + \alpha y_{t-1} + \varepsilon_t$. The test statistic we use as a feature is the usual KPSS statistic with the Bartlett window parameter set to the integer value of $4(T/100)^{0.25}$ (Pfaff 2008). This is the default value returned from the `ur.kpss()` function in the `urca` package (Pfaff, Zivot & Stigler 2016).

Autocorrelation coefficient based features

We calculate the first-order autocorrelation coefficient and the sum of squares of the first five autocorrelation coefficients of the original series, the first-differenced series, the second-differenced series, and the seasonally differenced series (for seasonal data).

A linear trend model is fitted to the time series, and the first-order autocorrelation coefficient of the residual series is calculated.

We calculate the sum of squares of the first five partial autocorrelation coefficients of the original series, the first-differenced series and the second-differenced series.

References

Armstrong, JS (2001). Should we redesign forecasting competitions? *International Journal of Forecasting* **17**(1), 542–543.

Breiman, L (2001). Random forests. *Machine Learning* **45**(1), 5–32.

Breiman, L & A Cutler (2004). *Random Forests*. Version 5.1. https://www.stat.berkeley.edu/~breiman/RandomForests/.

Breiman, L, A Cutler, A Liaw & M Wiener (2018). *randomForest: Breiman and Cutler's Random Forests for Classification and Regression*. R package version 4.6-14. https://cran.r-project.org/web/packages/randomForest/.

Chen, C, A Liaw & L Breiman (2004). *Using random forest to learn imbalanced data*. Tech. rep. University of California, Berkeley. http://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf.

Cleveland, RB, WS Cleveland & I Terpenning (1990). STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics* **6**(1), 3.

Collopy, F & JS Armstrong (1992). Rule-based forecasting: development and validation of an expert systems approach to combining time series extrapolations. *Management Science* **38**(10), 1394–1414.

Fraley, C (2012). *fracdiff: Fractionally differenced ARIMA aka ARFIMA(p,d,q) models*. R package version 1.4-2. https://CRAN.R-project.org/package=fracdiff.

Friedman, JH (1984). *A variable span scatterplot smoother*. Technical Report 5. Laboratory for Computational Statistics, Stanford University.

Fulcher, BD & NS Jones (2014). Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering* **26**(12), 3026–3037.

Goerg, GM (2016). *ForeCA: An R package for forecastable component analysis*. R package version 0.2.4. https://CRAN.R-project.org/package=ForeCA.

Goerg, G (2013). Forecastable component analysis. In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp.64–72.

Guerrero, VM (1993). Time-series analysis supported by power transformations. *Journal of Forecasting* **12**, 37–48.

Haslett, J & AE Raftery (1989). Space-time modelling with long-memory dependence: Assessing Ireland's wind power resource. *Applied Statistics* **38**(1), 1–50.

Hyndman, RJ & G Athanasopoulos (2018). *Forecasting: principles and practice*. 2nd ed. Melbourne, Australia: OTexts. https://OTexts.org/fpp2/.

Hyndman, RJ & Y Khandakar (2008). Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software* **26**(3), 1–22. http://www.jstatsoft.org/article/view/v027i03.

Hyndman, RJ, AB Koehler, JK Ord & RD Snyder (2008). *Forecasting with exponential smoothing: the state space approach*. Berlin: Springer.

Hyndman, RJ, AB Koehler, RD Snyder & S Grose (2002). A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting* **18**(3), 439–454.

Hyndman, RJ, E Wang & N Laptev (2015). Large-scale unusual time series detection. In: *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on*. IEEE, pp.1616–1619.

Hyndman, R, G Athanasopoulos, C Bergmeir, G Caceres, L Chhay, M O'Hara-Wild, F Petropoulos, S Razbash, E Wang & F Yasmeen (2018). *forecast: Forecasting functions for time series and linear models*. R package version 8.3. http://pkg.robjhyndman.com/forecast.

Kalousis, A & T Theoharis (1999). NOEMON: Design, implementation and performance results of an intelligent assistant for classifier selection. *Intelligent Data Analysis* **3**(5), 319–337.

Kück, M, SF Crone & M Freitag (2016). Meta-learning with neural networks and landmarking for forecasting model selection an empirical evaluation of different feature sets applied

to industry data. In: *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, pp.1499–1506.

Lemke, C & B Gabrys (2010). Meta-learning for time series forecasting and forecast combination. *Neurocomputing* **73**(10), 2006–2016.

Liaw, A & M Wiener (2002). Classification and regression by randomForest. *R News* **2**(3), 18–22. http://CRAN.R-project.org/doc/Rnews/.

Makridakis, S, A Andersen, R Carbone, R Fildes, M Hibon, R Lewandowski, J Newton, E Parzen & R Winkler (1982). The accuracy of extrapolation (time series) methods: results of a forecasting competition. *Journal of forecasting* **1**(2), 111–153.

Makridakis, S & M Hibon (2000). The M3-Competition: results, conclusions and implications. *International Journal of Forecasting* **16**(4), 451–476.

Nuttall, AH & GC Carter (1982). Spectral estimation using combined time and lag weighting. *Proceedings of the IEEE* **70**(9), 1115–1125.

Pfaff, B (2008). *Analysis of integrated and cointegrated time series with R*. Springer.

Pfaff, B, E Zivot & M Stigler (2016). *urca: Unit root and cointegration tests for time series data*. R package version 1.3-0. https://CRAN.R-project.org/package=urca.

Prudêncio, RB & TB Ludermir (2004). Meta-learning approaches to selecting time series models. *Neurocomputing* **61**, 121–137.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/.

Rice, JR (1976). The algorithm selection problem. *Advances in Computers* **15**, 65–118.

Shah, C (1997). Model selection in univariate time series forecasting using discriminant analysis. *International Journal of Forecasting* **13**(4), 489–500.

Smith-Miles, K (2009). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR)* **41**(1), 6.

Teräsvirta, T, CF Lin & CWJ Granger (1993). Power of the neural network linearity test. *Journal of Time Series Analysis* **14** (2), 209–220.

Wang, X, K Smith-Miles & RJ Hyndman (2009). Rule induction for forecasting method selection: meta-learning the characteristics of univariate time series. *Neurocomputing* **72**(10), 2581–2594.

Widodo, A & I Budi (2013). "Model selection using dimensionality reduction of time series characteristics". Paper presented at the International Symposium on Forecasting, Seoul, South Korea. June 2013. https://goo.gl/ig2J57.